La base de datos sakila de Oracle representa una tienda de alquiler de películas. En sakila vemos una tabla llamada *country* la cual cuenta con tres atributos: *country\_id, country* y *last\_update*. Se requiere juntar la información contenida en esta tabla con un JSON externo de países con información mas robusta como el indicador gini, los países fronterisos, los idiomas oficiales; en total cuenta con 24 atributos. El resultado de esta unión se almacenará en la tabla *country\_ext*. Parte del reto es mantener la integridad con las tablas transaccionales; la otra parte son las decisiones de diseño que hay que tomar, por ejemplo ¿Cómo representar los arreglos y los mapas en una base de datos relacional?

El proceso para el reto comienza con la instalación de un servidor MySQL local y MySQL Workbench como interfaz grafica para acceder al servidor. Luego pasamos a descargar la base de datos sakila, como todas las bases de datos relacionales cuenta con un esquema y un *query* para la inserción de los datos en la base de datos. En este punto pasamos a entender la base de datos, en este caso esta muy bien documentada, con imágenes del esquema y un diccionario de datos. Del diccionario vemos que *country\_id* es el consecutivo único que identifica los diferentes países, *country* es el nombre del país y *last\_update* es la fecha de la última modificación del registro. Para el entendimiento del JSON no contamos con esquemas o diccionario de datos, lo que puede dificultar la interpretación de algunos campos. Luego de entender el problema e identificar la forma de relacionar las fuentes de datos (campo *country* para sakila y *name* para JSON) pasamos a la creación de un script en Python apoyándonos en las bibliotecas SQLAlchemy para la conexión con mysql, pandas para estructurar los datos y json para el manejo de objetos JSON. Para abrir el archivo hay que especificar el tipo de *encoding* debido a que hay letras que se salen del alfabeto inglés.

A screenshot of a social media post

Description automatically generated

En esta implementación los arreglos y mapas del objeto JSON serán convertidos a una cadena de caracteres usando la función lambda para recorrer cada elemento de la columna y luego aplicarle *json.dumps()*. El formato JSON es muy fácil de reconstruir a partir del *string* por lo que no representa una perdida de información.

A screenshot of a cell phone

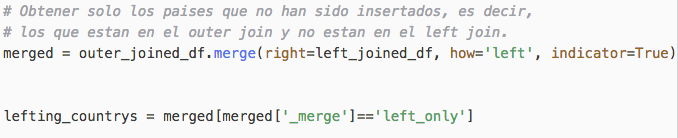
Description automatically generated

Para simplemente agregar los campos del JSON a los actuales países realizamos un *left-join*. Donde solo a los elementos en común se le agregarán los datos.

A close up of a sign

Description automatically generated

Luego necesitamos agregar los nuevos países al final de la lista, aunque esto dañe el orden alfabético de los datos. Esto permitirá mantener la consistencia de la base de datos intacta. Para este procedimiento hacemos un *outer\_join.* Obteniendo todos países del JSON y de la base de datos, más ambos atributos resultando en una tabla de 267 registros con 26 atributos. Para luego hacerle otro ­*left\_join* con la variable *left\_joined\_df* para obtener los datos que están en todo el conjunto menos los previamente unidos. Estos nuevos datos tienen los atributos de los paises en el JSON si coinciden en el nombre.



Esta ultima variable *lefting\_countrys* se logra al expresar en la variable *merged* el parametro *indicator=True*. Que quiere decir agregue una nueva columna indicando la procedencia de la fila. Para este caso, un *left join,* solo puede tener los valores *both* o *left\_only*. Seguido le decimos que solo obtenga los valores que en la columna nueva tengan el valor *left\_only* estos son los valores únicos en el JSON.

Finalmente agregamos los *country\_id* para los nuevos datos para ya hacer la inserción en la nueva tabla

A screenshot of a cell phone

Description automatically generated

